To our customers,

Old Company Name in Catalogs and Other Documents

On April 1st, 2010, NEC Electronics Corporation merged with Renesas Technology Corporation, and Renesas Electronics Corporation took over all the business of both companies. Therefore, although the old company name remains in this document, it is a valid Renesas Electronics document. We appreciate your understanding.

Renesas Electronics website: http://www.renesas.com

April 1st, 2010
Renesas Electronics Corporation

Issued by: Renesas Electronics Corporation (http://www.renesas.com)

Send any inquiries to http://www.renesas.com/inquiry.

Notice

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.
2. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.
4. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
5. When exporting the products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You should not use Renesas Electronics products or the technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations.
6. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
7. Renesas Electronics products are classified according to the following three quality grades: “Standard”, “High Quality”, and “Specific”. The recommended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below. You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application categorized as “Specific” without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for an application categorized as “Specific” or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is “Standard” unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.

   “Standard”: Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots.

   “High Quality”: Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; safety equipment; and medical equipment not specifically designed for life support.

   “Specific”: Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.
8. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

(Note 1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

RENESAS

Application Note

Multimedia Processor for Mobile Applications

UART Interface

EMMA Mobile1

Document No. **S19893EJ1V0AN00**
Date Published **AUG.2009**

PREFACE

Purpose The purpose of this document is to specify the usage of the UART interface.

Organization This document includes the following:

- Introduction
- Usage of UART Interface
- Example of UART Operation
- UART Driver Function

Notation Here explains the meaning of following words in text:

Note Explanation of item indicated in the text

Caution Information to which user should afford special attention

Remark Supplementary information

Related document The following tables list related documents.

Reference Document

Document Name	Version/date	Author	Description
S19265EJ1V0UM00_ASMUGIO.pdf	1st edition	NECEL	ASMU/GIO User's Manual
S19268EJ1V0UM00_1chip.pdf	1st edition	NECEL	1 Chip User's Manual
S19262EJ1V0UM00_UART.pdf	1st edition	NECEL	UART User's Manual
S19255EJ1V0UM00_DMA.PDF	1st edition	NECEL	DMA User's Manual
S19907EJ1V0AN00_GD.pdf	1st edition	NECEL	GD Spec

Disclaimers

- **The information contained in this document is subject to change without prior notice in the future. Refer to the latest applicable data sheet(s) and user's manual when designing a product for mass production.**
- No part of this document may be copied or reproduced in any form or by any means without the prior written consent of NEC Electronics. NEC Electronics assumes no responsibility for any errors that may appear in this document.
- NEC Electronics does not assume any liability for infringement of patents, copyrights or other intellectual property rights of third parties by or arising from the use of NEC Electronics products listed in this documents or any other liability arising from the use of such products. No license, express, implied or otherwise, is granted under any patents, copyrights or other intellectual property rights of NEC Electronics or others.
- Descriptions of circuits, software and other related information in this document are provided for illustrative purposes in semiconductor product operation and application examples. The incorporation of these circuits, software and information in the design of a customers' equipment shall be done under the full responsibility of the customer. NEC Electronics assume no responsibility for any losses incurred by customers or third parties arising from the use of these circuits, software and information.
- While NEC Electronics endeavors to enhance the quality, reliability and safety of NEC Electronics products, customers agree and acknowledge that possibility of defects thereof cannot be eliminated entirely. To minimize risks of damage to property or injury (including death) to persons arising from defects in NEC Electronics products, customers must incorporate sufficient safety measures in their design, such as redundancy, fire-containment and anti-failure features.

Note)

1. "NEC Electronics" as used in this document means NEC Electronics Corporation and also includes its majority-owned subsidiaries.
2. "NEC Electronics products" means any product developed or manufactured by or for NEC Electronics (as defined above)
3. All trademarks or registered trademarks are the property of their respective owners. Registered trademarks ® and trademarks™ are not noted in this document.

CONTENTS

Chapter 1 Introduction 8

1.1 Outline 8
1.2 Development Environment 8

Chapter 2 Usage of UART Interface 9

2.1 Overview Procedure of UART Operation 9
2.2 Detail Procedure of UART Operation 12
2.2.1 Send and Receive 13
2.2.2 Automatic Flow Control with CPU or DMA 16
2.2.3 Simulate Flow Control 20
2.2.4 Modem 23
2.2.5 IrDA 23

Chapter 3 Example of UART Operation 24

3.1 Initialization 25
3.1.1 Operation Flow 25
3.1.2 Operation Detail 26
3.2 Example of Send and Receive 28
3.2.1 Operation Flow 29
3.2.2 Operation Detail 30
3.3 Example of Automatic Flow Control 33
3.3.1 Operation Flow 34
3.3.2 Operation Detail 35
3.4 Example of Simulate Flow Control 37
3.4.1 Operation Flow 38
3.4.2 Operation Detail 39
3.5 Example of Automatic Flow Control with DMA 41
3.5.1 Operation Flow 42
3.5.2 Operation Detail 43

Appendix A UART Driver Function 45

A.1 UART Driver Function List 45
A.2 UART Global Variable Define 46
A.3 UART Structure Define 47
A.3.1 st_UART_SETTING 47
A.4 UART Driver Function Detail 48
A.4.1 Initialize 48
A.4.2 Set the Configure 51
A.4.3 Get the Configure 54
A.4.4 Send a Character 56
A.4.5 Receive a Character 58
A.4.6 Enable the Flow Control 59

A.4.7 Disable the Flow Control 61
A.4.8 Enable FIFO 62
A.4.9 Disable FIFO 65
A.4.10 Initialize GIO for Simulate Flow Control 66
A.4.11 Send a Character with Simulate Flow Control 67
A.4.12 Receive a Character with Simulate Flow Control 69
A.4.13 Clear the Receive Error 70
A.4.14 Print the Receive Error 71
A.4.15 Print the Configure 72
A.4.16 Initialize Port for Flow Control 73
A.4.17 Enable GIO for Receive 74
A.4.18 Enable UART DMA 75
A.4.19 Disable UART DMA 77

ANNEX Modification History 78

LIST OF TABLES

Table 1-1 Hardware Environment 8
Table 1-2 Software Environment 8
Table 2-1 Alternate Port 10
Table 2-2 UART Divisor Definition 14
Table 2-3 UART Transmit/Receive Bit Length Definition 15
Table 2-4 UART Parity Definition 15
Table 2-5 UART Stop Bit Definition 15
Table A-1 UART Driver Function List 45
Table A-2 Global Variable Define 46
Table A-3 Structure Define 47
Table A-4 Structure of st_UART_SETTING 47

LIST OF FIGURES

Figure 2-1 The Overview Operation 9
Figure 2-2 The Function of Send and Receive 13
Figure 2-3 The Data Format 14
Figure 2-4 The Function of Automatic Flow Control 16
Figure 2-5 The CTS Control Signal 19
Figure 2-6 The RTS Control Signal 19
Figure 2-7 The Function of Simulate Flow Control 20
Figure 2-8 The Simulate CTS Control Signal 22
Figure 2-9 The Simulate RTS Control Signal 22
Figure 3-1 UART0 Initialization 25
Figure 3-2 Hardware Connection of Send and Receive 28
Figure 3-3 Example of Send and Receive 29
Figure 3-4 Hardware Connection of Automatic Flow Control 33
Figure 3-5 Example of Automatic Flow Control 34
Figure 3-6 Hardware Connection of Simulate Flow Control 37
Figure 3-7 Example of Simulate Flow Control 38
Figure 3-8 Example of Automatic Flow Control with DMA 42
Figure A-1 UART Initialize 49
Figure A-2 Set the Configure 52
Figure A-3 Get the Configure 54
Figure A-4 Send a Character 57
Figure A-5 Receive a Character 58
Figure A-6 Enable Flow Control 59
Figure A-7 Disable Flow Control 61

Figure A-8 Enable FIFO 63
Figure A-9 Disable FIFO 65
Figure A-10 Send a Character with Simulate Flow Control 67
Figure A-11 Received a Character with Simulate Flow Control 69
Figure A-12 Clear the Receive Error 70
Figure A-13 Print the Receive Error 71
Figure A-14 Enable UART DMA 75
Figure A-15 Disable UART DMA 77

Chapter 1 Introduction

1.1 Outline

This document introduces the Universal Asynchronous Receiver/Transmitter (UART) of EMMA Mobile1.

- How to send and receive characters.
- How to send and receive characters with automatic flow control.
- How to send and receive characters with simulate flow control.
- How to send and receive characters with DMA.

The UART block incorporated in EMMA Mobile1 has two 64-byte FIFO buffers, one for transmission and one for reception.
About the details of functions please refer to "**CHAPTER 1 OVERVIEW**" of EMMA Mobile1 UART user's manual.

1.2 Development Environment

- Hardware environment of this project is listed as below.

Table 1-1 Hardware Environment

Name	Version	Maker
EMMA Mobile1 evaluation board (PSKCH2Y-S-0016-01)	-	NEC Electronics
PARTNER-Jet ICE ARM	M20	Kyoto Microcomputer Co. Ltd

- Software used in this project is listed as below.

Table 1-2 Software Environment

Name	Version	Maker
GNUARM Toolchain	V4.3.2	GNU
WJETSET-ARM	V5.10a	Kyoto Microcomputer Co. Ltd

Chapter 2 Usage of UART Interface

2.1 Overview Procedure of UART Operation

The following picture is the overview operation process for UART.

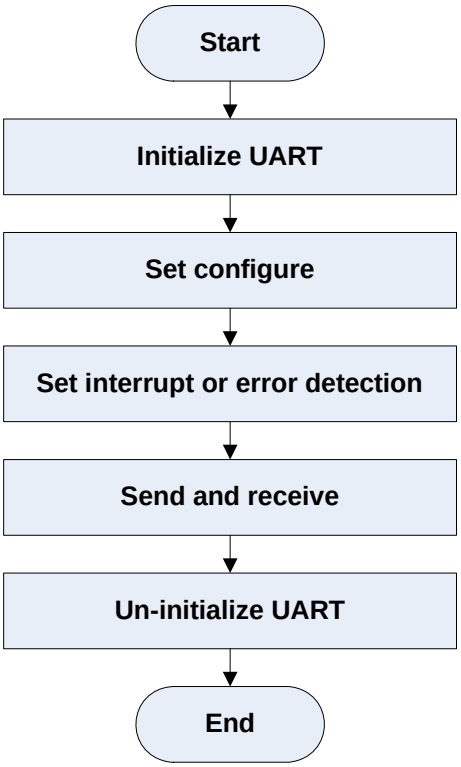

Figure 2-1 The Overview Operation

(1). Initialize the UART.

- Open the clock

The related registers are as follow:

ASMU_GCLKCTRL2ENA;
ASMU_GCLKCTRL2;
ASMU_DIVU70SCLK;
ASMU_DIVU71SCLK;
ASMU_DIVU72SCLK;

- Reset the UART

The related registers are as follow:

ASMU_RESETREQ0ENA;

ASMU_RESETCTRL0;

- Port setting

The related registers are as follow:

For port setting:

CHG_PINSEL_G80;

CHG_PINSEL_G96;

For input attribute setting:

CHG_PULL0;

CHG_PULL_G80;

CHG_PULL_G104;

For driver capability setting:

CHG_DRIVE2;

For wait control and read mode:

ASMU_AB1_U70WAITCTRL;

ASMU_AB1_U71WAITCTRL;

ASMU_AB1_U72WAITCTRL;

ASMU_AB1_U70READCTRL;

ASMU_AB1_U71READCTRL;

ASMU_AB1_U72READCTRL;

- Disable UART interrupt

The related registers are as follow:

SEC_IT0_IDSS0;

INTC_IT0_IDS0;

About the details, please refer to the "**Appendix A.4.1 Initialize**" of this document.

For port setting, the following table is the alternate port table:

Table 2-1 Alternate Port

Port Name	Function1	Function2	Function3
URT0_CTSB	GIO_P85	URT0_CTSB	URT1_SRIN
URT0_RTSB	GIO_P86	URT0_RTSB	URT1_SOUT
URT2_SRIN	GIO_P108	URT2_SRIN	NAND_ALE
URT2_SOUT	GIO_P109	URT2_SOUT	NAND_CLE
URT2_CTSB	GIO_P110	URT2_CTSB	NAND_D0
URT2_RTSB	GIO_P111	URT2_RTSB	NAND_D1

(2). Set the configuration

- Attribute setting

The related registers are as follow (**Note:** x is 0, 1 or 2):

EM1_UARTx_LCR;
EM1_UARTx_DLM;
EM1_UARTx_DLL;

- FIFO setting

The related registers are as follow:

EM1_UARTx_FCR;

- Flow control setting

The related registers are as follow:

EM1_UARTx_HCR0;
EM1_UARTx_MCR;

(3). Set the interrupt or error detection.

The related registers are as follow:

EM1_UARTx_IER;
SEC_IT0_IENS0;
INTC_IT0_IEN0;

(4). Send and receive.

The related registers are as follow:

EM1_UARTx_LSR
EM1_UARTx_THR
EM1_UARTx_RBR

(5). Un-initialize the UART.

- FIFO setting

The related registers are as follow:

EM1_UARTx_FCR;

- Flow control setting

The related registers are as follow:

EM1_UARTx_HCR0;
EM1_UARTx_MCR;

2.2 Detail Procedure of UART Operation

According to the hardware feature, the UART can have the following function:

(1). Send and receive characters.

(2). Automatic flow control with CPU or DMA.

(3). Simulate flow control.

(4). Modem.

(5). IrDA.

Note: the above process does not have initialization. About the initialization, please refer to the "**Chapter 2.1 Overview Procedure of UART Operation**" of this document.

2.2.1 Send and Receive

The following figure shows the operation process for sending and receiving without flow control.

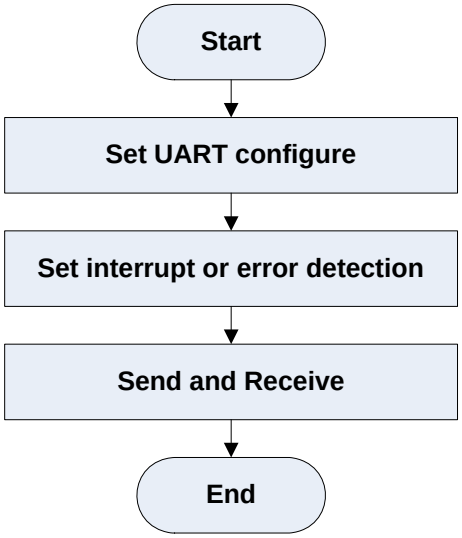

Figure 2-2 The Function of Send and Receive

(1). Set the configuration.

Set the baud rate, data length, parity, stop bit, error detection, and disable the flow control. The related registers are as follow (**Note:** x is 0, 1 or 2):

EM1_UARTx_LCR;
EM1_UARTx_DLM;
EM1_UARTx_DLL;

(2). Set the interrupt or error detection.

The related registers are as follow:

EM1_UARTx_IER;
SEC_IT0_IENS0;
INTC_IT0_IEN0;

(3). Send and receive.

The related registers are as follow:

EM1_UARTx_LSR;
EM1_UARTx_THR;
EM1_UARTx_RBR;

- Data Format

The UART transmit/receive format is as following figure:

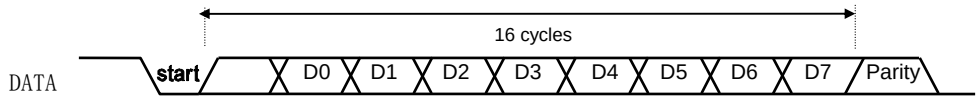

Figure 2-3 The Data Format

After one start bit, 5-8 data bits, then one optional parity bit. At last, one or two stop bit.

- Baud rate

The EM1_UARTx_DLM register and EM1_UARTx_DLL register control the divisor of EMMA Mobile1 UART. About the bit definition, please refer to the "**Chapter 3.2.10 Divisor latch LS byte register**" and "**Chapter 3.2.11 Divisor latch MS byte register**" of EMMA Mobile1 UART user's manual.

The formula of divisor is as follow:
Divisor set value = reference clock frequency [Hz] / (requested baud rate [bps] × 16);
The reference clock frequency is PLL3. If it is 229.376MHz, the DLM and DLL setting values are as following table:

Table 2-2 UART Divisor Definition

Baud rate	DLM	DLL
300	0x00BA	0x00AB
600	0x005D	0x0055
1200	0x002E	0x00AB
2400	0x0017	0x0055
4800	0x000B	0x00AB
9600	0x0005	0x00D5
14400	0x0003	0x00E4
19200	0x0002	0x00EB
38400	0x0001	0x0075
57600	0x0000	0x00F9
115200	0x0000	0x007C
230400	0x0000	0x003E
460800	0x0000	0x001F
4000000	0x0000	0x0004

About the divisor setting, please refer to the "**Chapter 5.2 Baud Rate Setting**" of EMMA Mobile1 UART user's manual.

- Bit Length

The bit[1:0] of EM1_UARTx_LCR register control the transmit/receive bit length of EMMA Mobile1 UART. About the bit definition, please refer to the "**Chapter 3.2.5 Line control register**" of EMMA Mobile1 UART user's manual.

The bit length setting is as following table:

Table 2-3 UART Transmit/Receive Bit Length Definition

Bit Length	Bit 1	Bit 0
5 bit length	0	0
6 bit length	0	1
7 bit length	1	0
8 bit length	1	1

- Parity

The bit[5:3] of EM1_UARTx_LCR register control the parity of EMMA Mobile1 UART. About the bit definition, please refer to the "**Chapter 3.2.5 Line control register**" of EMMA Mobile1 UART user's manual.

The parity setting is as following table:

Table 2-4 UART Parity Definition

Parity Type	Bit 5	Bit 4	Bit 3
Non parity	x	x	0
Odd parity	0	0	1
Even parity	0	1	1
Stick high (fixed to 1)	1	0	1
Stick low (fixed to 0)	1	1	1

Note: x - don't care.

- Stop Bit

The bit[2] of EM1_UARTx_LCR register control the stop bit numbers of EMMA Mobile1 UART. About the bit definition, please refer to the "**Chapter 3.2.5 Line control register**" of EMMA Mobile1 UART user's manual.

The stop bit setting is as following table:

Table 2-5 UART Stop Bit Definition

Stop Bit Numbers	Bit 2
1	0
2	1

2.2.2 Automatic Flow Control with CPU or DMA

The following figure shows the operation process for sending and receiving with automatic flow control.

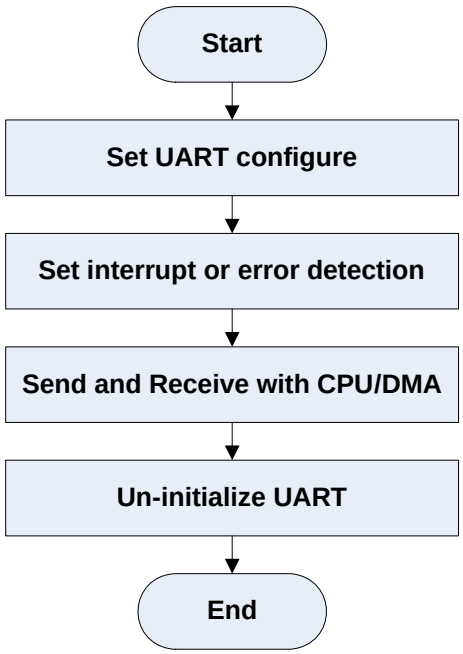

Figure 2-4 The Function of Automatic Flow Control

(1). Set the configuration.

Set the baud rate, data length, parity, stop bit, error detection. If enable flow control, set the FIFO and enable flow control.

- Attribute setting

The related registers are as follow (**Note:** x is 0, 1 or 2):

EM1_UARTx_LCR;
EM1_UARTx_DLM;
EM1_UARTx_DLL;

- FIFO setting

The related registers are as follow:

EM1_UARTx_FCR;

- Flow control setting

The related registers are as follow:

EM1_UARTx_HCR0;
EM1_UARTx_MCR;

(2). Set the interrupt or error detection.

The related registers are as follow:

EM1_UARTx_IER;

SEC_IT0_IENS0;
INTC_IT0_IEN0;

(3). Send and receive.
The related registers are as follow:
EM1_UARTx_LSR
EM1_UARTx_THR
EM1_UARTx_RBR

When use DMA, the registers of DMA are as follow:
DMA_M2P_CONT;
DMA_M2P_PE0_LCH0LCH3_INT_RAW

DMA_M2P_LCH0_AADD;
DMA_M2P_LCH0_AOFF;
DMA_M2P_LCH0_ASIZE;
DMA_M2P_LCH0_ASIZE_COUNT;
DMA_M2P_LCH0_BADD;
DMA_M2P_LCH0_LENG;
DMA_M2P_LCH0_MODE;

DMA_P2M_LCH0_AADD;
DMA_P2M_LCH0_BOFF;
DMA_P2M_LCH0_BSIZE;
DMA_P2M_LCH0_BSIZE_COUNT;
DMA_P2M_LCH0_BADD;
DMA_P2M_LCH0_LENG;
DMA_P2M_LCH0_MODE;

DMA_M2P_LCH1_AADD;
DMA_M2P_LCH1_AOFF;
DMA_M2P_LCH1_ASIZE;
DMA_M2P_LCH1_ASIZE_COUNT;
DMA_M2P_LCH1_BADD;
DMA_M2P_LCH1_LENG;
DMA_M2P_LCH1_MODE;

DMA_P2M_LCH1_AADD;
DMA_P2M_LCH1_BOFF;
DMA_P2M_LCH1_BSIZE;
DMA_P2M_LCH1_BSIZE_COUNT;
DMA_P2M_LCH1_BADD;
DMA_P2M_LCH1_LENG;

DMA_P2M_LCH1_MODE;

DMA_M2P_LCH2_AADD;
DMA_M2P_LCH2_AOFF;
DMA_M2P_LCH2_ASIZE;
DMA_M2P_LCH2_ASIZE_COUNT;
DMA_M2P_LCH2_BADD;
DMA_M2P_LCH2_LENG;
DMA_M2P_LCH2_MODE;

DMA_P2M_LCH2_AADD;
DMA_P2M_LCH2_BOFF;
DMA_P2M_LCH2_BSIZE;
DMA_P2M_LCH2_BSIZE_COUNT;
DMA_P2M_LCH2_BADD;
DMA_P2M_LCH2_LENG;
DMA_P2M_LCH2_MODE;

About how to use DMA, please refer to the EMMA Mobile1 DMA user's manual.

(4). Un-initialize the UART.
Un-initialization only disables FIFO and flow control.

- FIFO setting

The related registers are as follow:
EM1_UARTx_FCR;

- Flow control setting

The related registers are as follow:
EM1_UARTx_HCR0;
EM1_UARTx_MCR;

The setting of baud rate, bit numbers, parity and stop bit is the same to "**Chapter 2.2.1 Send and Receive**" of this document, so about the details, please refer to that chapter.

- CTS control signal

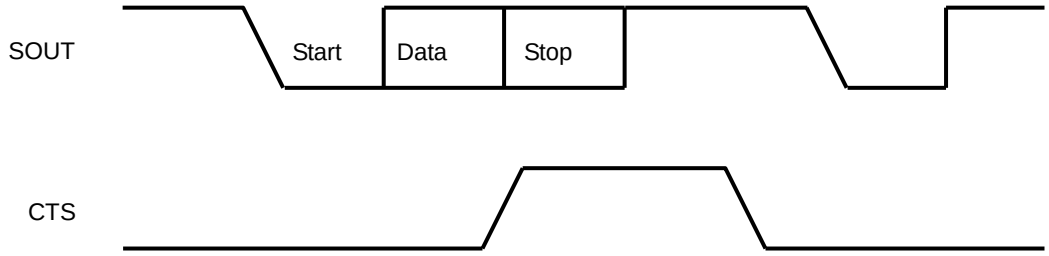

Figure 2-5 The CTS Control Signal

When the CTS pin is set to the low level (transmit request), data in the transmit buffer is transmitted.
When the CTS pin changes from the low level to the high level during data transmission, data being transmitted is transmitted fully and transmit of the next data is stopped.

- RTS control signal

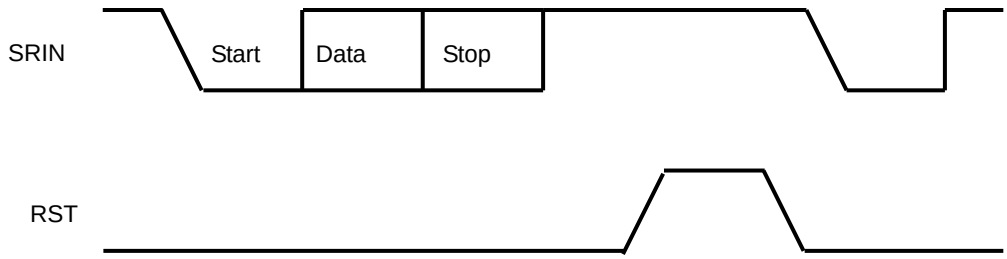

Figure 2-6 The RTS Control Signal

The FIFO is hardware to send and receive characters automatically.
When the number of data in the receive FIFO reaches the trigger level, the RTSZ pin is set to the high level (transmit stop request).
When the data in the receive FIFO is read, it becomes empty or has less data than the trigger level, the RTSZ pin is set to the low level (transmit request).

2.2.3 Simulate Flow Control

The following figure shows the operation process for sending and receiving with simulate flow control.

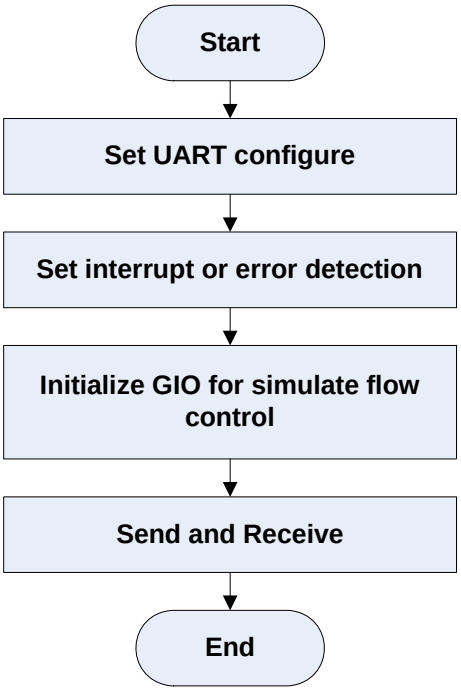

Figure 2-7 The Function of Simulate Flow Control

(1). Set the configuration.

Set the baud rate, data length, parity, stop bit, error detection, and disable the flow control.

The related registers are as follow (**Note:** x is 0, 1 or 2):

EM1_UARTx_LCR;

EM1_UARTx_DLM;

EM1_UARTx_DLL;

(2). Set the interrupt or error detection.

The related registers are as follow:

EM1_UARTx_IER;

SEC_IT0_IENS0;

INTC_IT0_IEN0;

(3). Initialize the GIO for simulate flow control.

In this document, only use GIO85 for CTS and GIO86 for RTS.

The related registers are as follow:

CHG_PINSEL_G80;

GIO_E1_HH; GIO_E0_HH;

CHG_PULL_G80

(4). Send and receive.

The related registers are as follow:

EM1_UARTx_LSR

EM1_UARTx_THR

EM1_UARTx_RBR

The setting of baud rate, bit numbers, parity and stop bit is the same to "**Chapter 2.2.1 Send and Receive**" of this document, so about the details, please refer to that chapter.

- GIO85 simulates CTS control signal

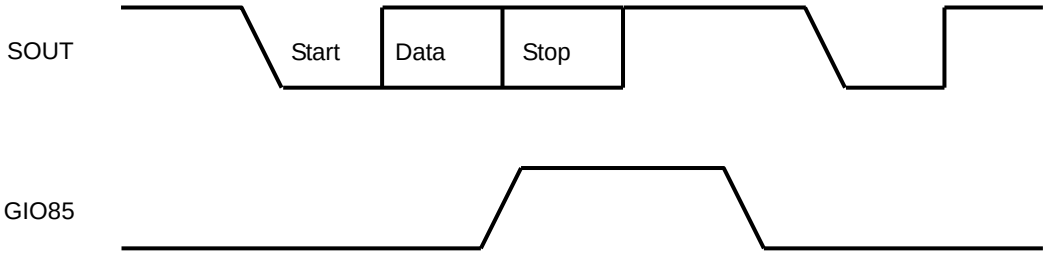

Figure 2-8 The Simulate CTS Control Signal

When the GIO85 pin is set to the low level (transmit request), data can be transmitted.
When the GIO85 pin changes from the low level to the high level during data transmission, data being transmitted is transmitted fully and transmit of the next data is stopped.

- GIO86 simulates RTS control signal

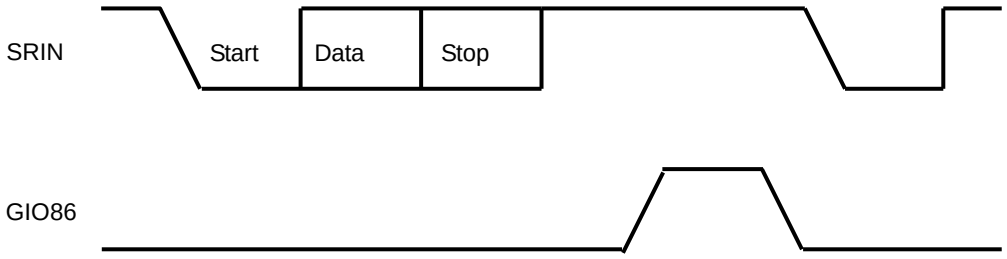

Figure 2-9 The Simulate RTS Control Signal

When the EMMA Mobile1 can receive characters, the GIO86 pin is set to the low level (transmit request).
When the EMMA Mobile1 can not receive characters, the GIO86 pin is set to the high level (transmit stop request).

2.2.4 Modem

About the Modem, please refer to the “**Chapter 2.2.2 Automatic Flow Control**” of this document. And there is no example about this setting in the chapter 3.

2.2.5 IrDA

About the IrDA, please refer to the “**Chapter 2.2.2 Automatic Flow Control**” of this document. And there is no example about this setting in the chapter 3.

Chapter 3 Example of UART Operation

The following contents show 4 examples: how to send and receive, how to enable automatic flow control, how to use GIO to simulate flow control and how to use DMA to transmission. About the API details, please refer to the “**Appendix A UART Driver Function**”.

3.1 Initialization

3.1.1 Operation Flow

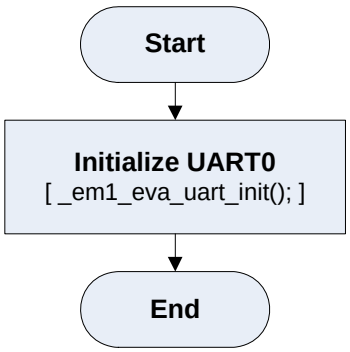

Figure 3-1 UART0 Initialization

About the UART function, please refer to the “**Appendix A UART Driver Function**”.

3.1.2 Operation Detail

(1). Initialize the UART0.
The process calls the em1_uart_init().
The em1_uart_init() finishes the following functions for UART:

- Open U70_CLK.

```
ASMU_GCLKCTRL2ENA[5] = 1b;
ASMU_GCLKCTRL2[5] = 1b;
ASMU_GCLKCTRL2ENA[5] = 0b;
```

- Open U70_SCLK.

```
ASMU_GCLKCTRL2ENA[6] = 1b;
ASMU_GCLKCTRL2[6] = 1b;
ASMU_GCLKCTRL2ENA[6] = 0b;
```

- Reset UART0.

```
ASMU_RESETREQ0ENA[27] = 1b;
ASMU_RESETREQ0[27] = 0b;
ASMU_RESETREQ0ENA[27] = 0b;
```

- Reset release UART0.

```
ASMU_RESETREQ0ENA[27] = 1b;
ASMU_RESETREQ0[27] = 1b;
ASMU_RESETREQ0ENA[27] = 0b;
```

- Set divisor.

```
ASMU_DIVU70SCLK = Divisor;
```

Divisor is an input value, about how to set the divisor, please refers to the "**Chapter 3.2.52 U70_SCLK frequency division setting register**" of EMMA Mobile1 ASMU/GIO user's manual.

- Switch port function.

```
CHG_PINSEL_G80[11:10] = 01b;        // GIO_P85 --> URT0_CTSB
CHG_PINSEL_G80[13:12] = 01b;        // GIO_P86 --> URT0_RTSB
```

- Enable input and disable pull up/down.

```
CHG_PULL0[30:28] = 111b;            // For URT0_SRIN.
CHG_PULL_G80[22:20]= 111b;          // For URT0_CTS.
```

- Set the driver capability.

```
CHG_DRIVE2[5:4] = 01b;              // 4mA
```

- Set the wait control and read mode.

```
ASMU_AB1_U70WAITCTRL = 0xF1F0F;
ASMU_AB1_U70READCTRL = 0;
```

- Disable the UART global interrupt.

```
SEC_IT0_IDSS0[9] = 1b;
INTC_IT0_IDS0[9] = 1b;
```

3.2 Example of Send and Receive

The hardware connection of send and receive is as follow figure.

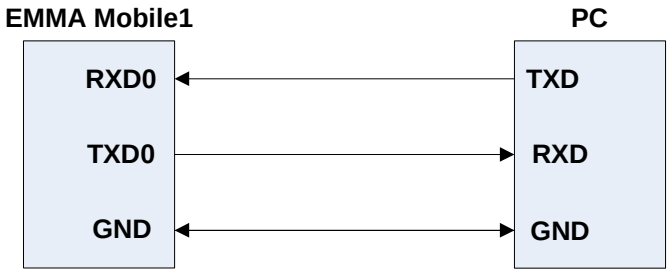

Figure 3-2 Hardware Connection of Send and Receive

3.2.1 Operation Flow

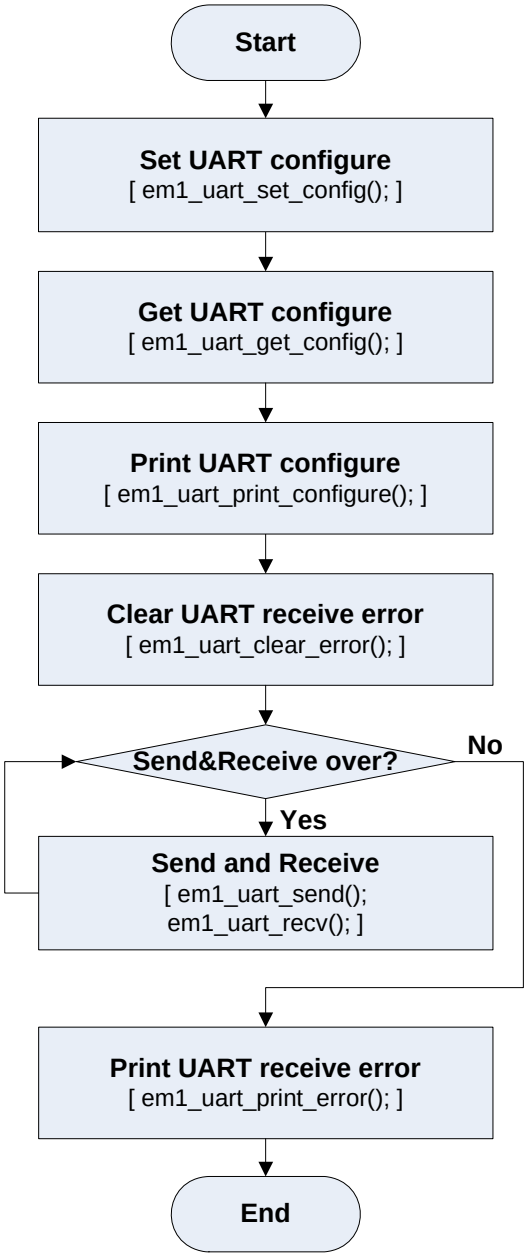

Figure 3-3 Example of Send and Receive

About the UART function, please refer to the "**Appendix A UART Driver Function**".

3.2.2 Operation Detail

(1). Set the UART configure.
The process calls the “em1_uart_set_config()” function.
The “em1_uart_set_config()” function finishes the following functions for UART0:

- Set parity.
  EM1_UART0_LCR[5:3] = 000b; // none parity.

- Set stop bit.
  EM1_UART0_LCR[2] = 0b; // 1 stop bit.

- Set length.
  EM1_UART0_LCR[1:0] = 11b; // 8 bits.

- Set divisor.
  EM1_UART0_LCR[7] = 1b;
  EM1_UART0_DLM = 0;
  EM1_UART0_DLL = 0x7C; // 115200 bps.
  EM1_UART0_LCR[7] = 0b;

- Enable receive error.
  EM1_UART0_IER = 0x4;

- Disable FIFO.
  EM1_UARTx_FCR[0] = 0b;

- Disable RTS and CTS.
  EM1_UARTx_MCR[5] = 0b; // Disable the automatic flow control.
  EM1_UARTx_MCR[1] = 1b; // Enable RTS.

(2). Get UART configure.
The process calls the “em1_uart_get_config()” function.
The “em1_uart_get_config()” function finishes the following functions for UART0:

- Get parity.
  parity = EM1_UART0_LCR[5:3];

- Get stop bit.
  stop-bit = EM1_UART0_LCR[2];

- Get length bit.

  length = EM1_UART0_LCR[1:0];

- Get divisor.

  DLM = EM1_UART0_DLM;

  DLL = EM1_UART0_DLL;

- Get flow control.

  mode = EM1_UART0_MCR[5];

(3). Print UART configure.
Print the information of baud rate, bit numbers, parity, stop bit and flow control.
And this function is used for giving more information about the related UART.

(4). Clear the receive error.
The process calls the “em1_uart_clear_error()” function.
The “em1_uart_clear_error()” function finishes the following functions for UART0:

- Read LSR register to clear error information.

  Error = EM1_UART0_LSR;

(5). UART send and receive.
The process calls the “em1_uart_recv()” function.
The “em1_uart_recv()” function finishes the following functions for UART0:

- Check receive finish.

  If EM1_UART0_LSR[0] is 1, receive finish.

- Read the received character.

  Data = EM1_UART0_RBR;

The process calls the “em1_uart_ send()” function.
The “em1_uart_ send()” function finishes the following functions for UART0:

- Check send ready.

  If EM1_UART0_LSR[5] is 1, send ready.

- Write the character to be sent.

  EM1_UART0_THR = data;

(6). Print the receive error.

The process calls the “em1_uart_print_error()” function.

The “em1_uart_print_error()” function finishes the following functions for UART0:

- Read the receive error.

  Error = EM1_UART0_LSR;

3.3 Example of Automatic Flow Control

The hardware connection of automatic flow control is as follow figure.

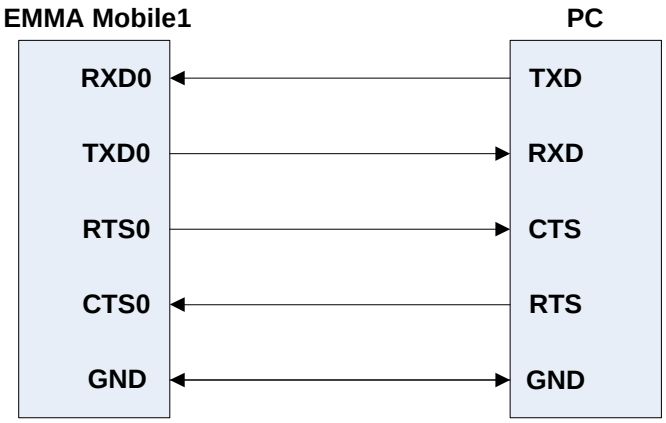

Figure 3-4 Hardware Connection of Automatic Flow Control

3.3.1 Operation Flow

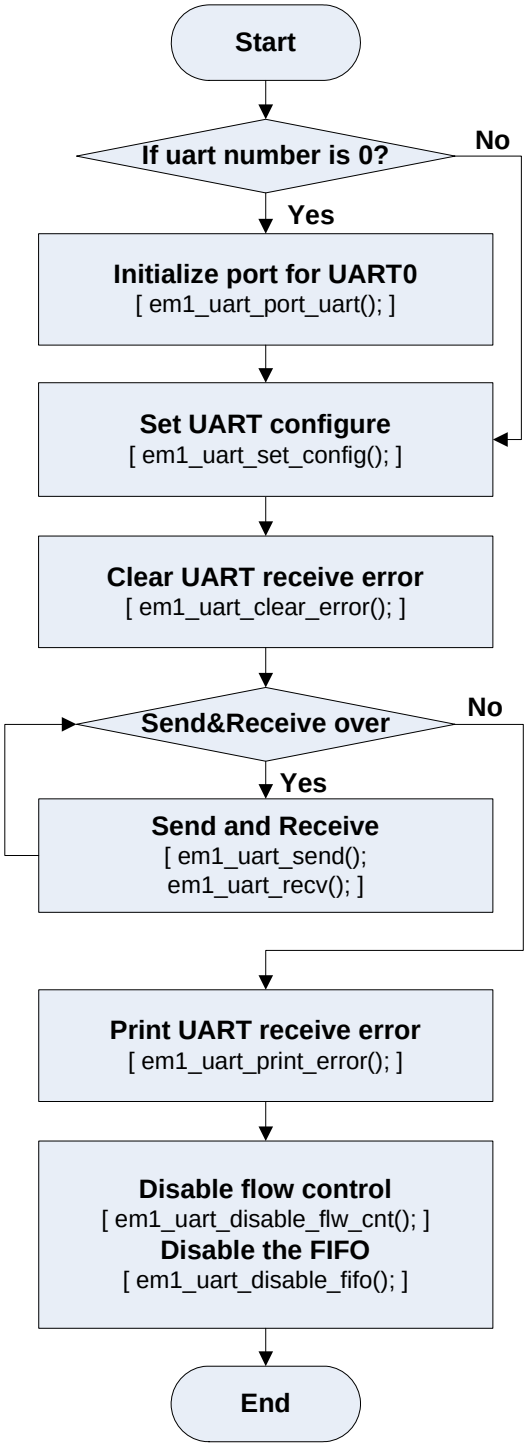

Figure 3-5 Example of Automatic Flow Control

About the UART function, please refer to the "**Appendix A UART Driver Function**".

3.3.2 Operation Detail

(1). Initialize the port for UART0.
The process calls the “em1_uart_port_uart()” function.
The “em1_uart_port_uart()” function finishes the following functions for UART0:

- Change the GIO_P85 to URT0_CTSB.
- Change the GIO_P86 to URT0_RTSB.
- Enable the URT0_CTSB input.

About the details, please refer to the ASMU/GIO user’s manual and 1 chip user’s manual.

(2). Set the UART configure.
The process calls the “em1_uart_set_config()” function.
The “em1_uart_set_config()” function finishes the following functions for UART0:

- Set parity.
  EM1_UART0_LCR[5:3] = 000b; // none parity.

- Set stop bit.
  EM1_UART0_LCR[2] = 0b; // 1 stop bit.

- Set length.
  EM1_UART0_LCR[1:0] = 11b; // 8 bits.

- Set divisor.
  EM1_UART0_LCR[7] = 1b;
  EM1_UART0_DLM = 0;
  EM1_UART0_DLL = 0x7C; // 115200 bps.
  EM1_UART0_LCR[7] = 0b;

- Enable receive error.
  EM1_UART0_IER = 0x4;

- Enable FIFO.
  EM1_UART0_FCR[2] = 1b; // Reset the transfer buffer of FIFO.
  EM1_UART0_FCR[1] = 1b; // Reset the receive buffer of FIFO.
  EM1_UART0_FCR[5] = 0b; // Set 16 bytes FIFO mode.
  EM1_UART0_FCR[7:6] = 00b; // The trigger number is 1.
  EM1_UART0_FCR[0] = 1b; // Enable FIFO.

- Enable RTS and CTS.
  EM1_UART0_HCR0[6] = 0b; // Set RTS mode 0.

```
EM1_UART0_MCR[5] = 1b;          // Enable the automatic flow control.
EM1_UART0_MCR[1] = 1b;          // Enable RTS.
```

(3). Clear the receive error.
This process is the same to process (4) of "**Chapter 3.2.2 Operation Detail**" of this document.

(4). UART send and receive.
This process is the same to process (5) of "**Chapter 3.2.2 Operation Detail**" of this document.

(5). Print the receive error.
This process is the same to process (6) of "**Chapter 3.2.2 Operation Detail**" of this document.

(6). Disable flow control and FIFO.
The process calls the "em1_uart_disable_flw_cnt()" function.
The "em1_uart_disable_flw_cnt()" function finishes the following functions for UART0:

- Disable RTS and CTS.

```
EM1_UARTx_MCR[5] = 0b;          // Disable the automatic flow control.
EM1_UARTx_MCR[1] = 1b;          // Enable RTS.
```

The process calls the "em1_uart_disable_fifo()" function.
The "em1_uart_disable_fifo()" function finishes the following functions for UART0:

- Disable FIFO.

```
EM1_UARTx_FCR[0] = 0b;
```

3.4 Example of Simulate Flow Control

The hardware connection of simulate flow control is as follow figure.

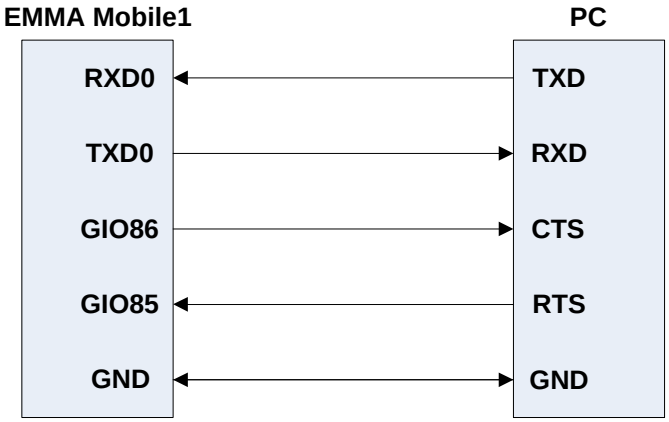

Figure 3-6 Hardware Connection of Simulate Flow Control

3.4.1 Operation Flow

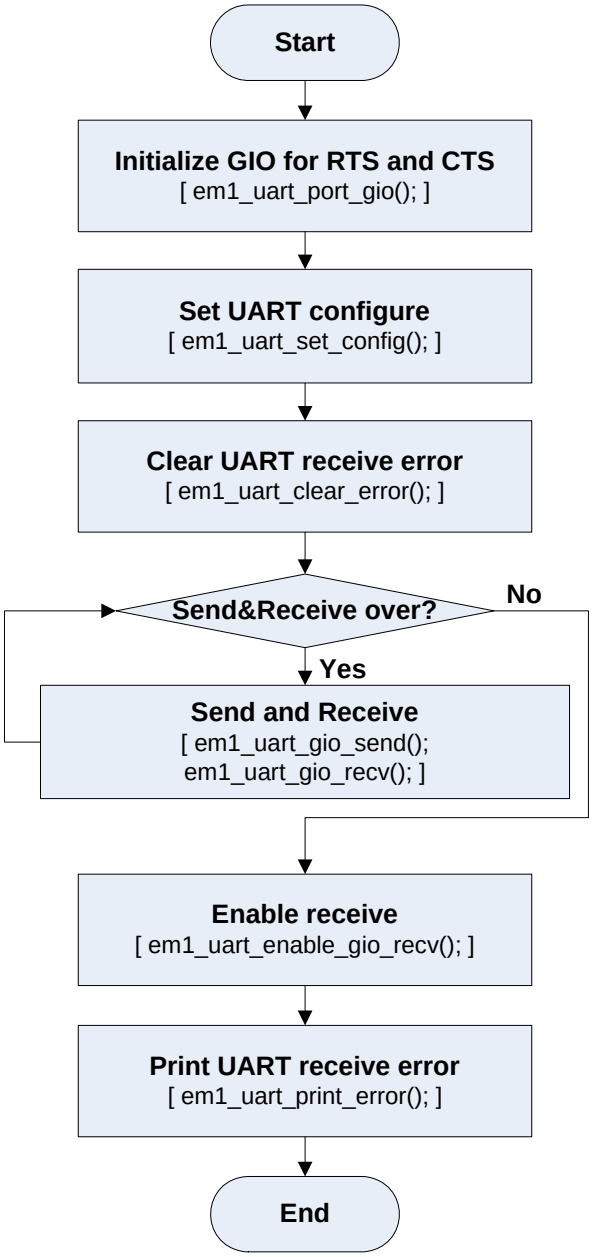

Figure 3-7 Example of Simulate Flow Control

About the UART function, please refer to the "**Appendix A UART Driver Function**".

3.4.2 Operation Detail

(1). Initialize the GIO85 and GIO86.
The process calls the “em1_uart_port_gio()” function.
The “em1_uart_port_gio()” function finishes the following functions for UART0:

- Change the URT0_RTSB to GIO_P86;
- Set the GIO_P86 output mode;
- Set the GIO_P86 output high level;
- Change the URT0_CTSB to GIO_P85;
- Set the GIO_P85 input mode;
- Enable the GIO_P85 input;

(2). Set the UART configure.
This process is the same to process (1) of “**Chapter 3.2.2 Operation Detail**” of this document.

(3). Clear the receive error.
This process is the same to process (4) of “**Chapter 3.2.2 Operation Detail**” of this document.

(4). UART send and receive.
The process calls the “em1_uart_gio_recv()” function.
The “em1_uart_gio_recv()” function finishes the following functions for UART0:

- Output low level from GIO86;
- Check whether receive a character;
- Read the received character;
- Output high level from GIO86;

The process calls the “em1_uart_gio_send()” function.
The “em1_uart_gio_send()” function finishes the following functions for UART0:

- Check if GIO85 is high or low;
- Check if there is a space for a character;
- Write the character to be send;

About the GIO details, please refer to the ASMU/GIO user’s manual and 1 chip user’s manual.
About the UART details, please refer to the process (5) of “**Chapter 3.2.2 Operation Detail**” o this document.

(5). Enable receive.
The process calls the “em1_uart_enable_gio_recv()” function.
The “em1_uart_enable_gio_recv()” function finishes the following functions for UART0:

- Output low level from GIO86;

About the details, please refer to the EMMA Mobile1 ASMU/GIO user’s manual and 1 chip user’s manual.

(6). Print the receive error.
This process is the same to process (6) of “**Chapter 3.2.2 Operation Detail**” of this document.

3.5 Example of Automatic Flow Control with DMA

About the hardware connection of automatic flow control, please refer to figure3-4.

3.5.1 Operation Flow

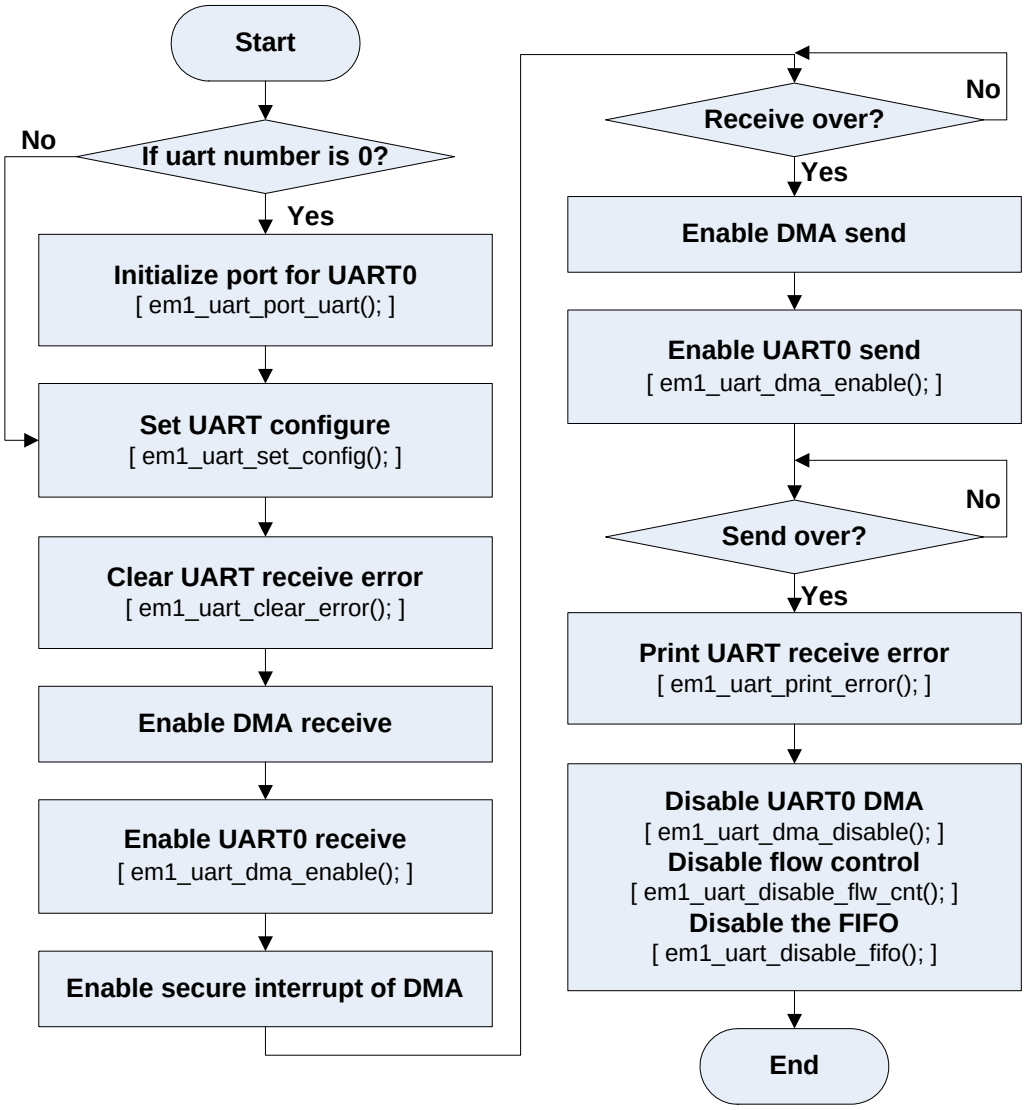

Figure 3-8 Example of Automatic Flow Control with DMA

About the UART function, please refer to the "**Appendix A UART Driver Function**".

3.5.2 Operation Detail

(1). Initialize the port for UART0.
This process is the same to process (1) of "**Chapter 3.3.2 Operation Detail**" of this document.

(2). Set the UART configure.
This process is the same to process (2) of "**Chapter 3.3.2 Operation Detail**" of this document.

(3). Clear the receive error.
This process is the same to process (4) of "**Chapter 3.2.2 Operation Detail**" of this document.

(4). Enable DMA receive.
About the details, please refer to the EMMA Mobile1 DMA user's manual.

(5). Enable UART0 receive.
The process calls the "em1_uart_dma_enable()" function.
The "em1_uart_dma_enable()" function finishes the following functions for UART0:

- DMA send mode0 and DMA receive mode0.
  EM1_UART0_HCR0[2] = 0b;
  EM1_UART0_HCR0[3] = 0b;
  EM1_UART0_FCR[3] = 0b;

- Exclude timeout.
  EM1_UART0_HCR0[4] = 1b;

- Access 1 byte.
  EM1_UART0_HCR0[5] = 1b;

- Enable DMA receive.
  EM1_UART0_HCR0[1] = 1b;

(6). Enable secure interrupt of DMA
SEC_IT0_IENS1[5] = 1;

(7). Check receive over.
If DMA_P2M_PE0_LCH0LCH3_INT_RAW[4] is 1, receive over.

(8). Enable DMA send.
About the details, please refer to the EMMA Mobile1 DMA user's manual.

(9). Enable UART0 send.
The process calls the “em1_uart_dma_enable()” function.
The “em1_uart_dma_enable()” function finishes the following functions for UART0:

- DMA send mode0 and DMA receive mode0.
  EM1_UART0_HCR0[2] = 0b;
  EM1_UART0_HCR0[3] = 0b;
  EM1_UART0_FCR[3] = 0b;

- Exclude timeout.
  EM1_UART0_HCR0[4] = 1b;

- Access 1 byte.
  EM1_UART0_HCR0[5] = 1b;

- Enable DMA send.
  EM1_UART0_HCR0[0] = 1b;

(10). Check receive over.
If DMA_M2P_PE0_LCH0LCH3_INT_RAW[4] is 1, receive over.

(11). Print the receive error.
This process is the same to process (6) of “**Chapter 3.2.2 Operation Detail**” of this document.

(12). Disable UART0 DMA, flow control and FIFO.
To disable UART0 DMA, call the “em1_uart_dma_disable()” function.
The “em1_uart_dma_disable()” function finishes the following functions for UART0:

- Disable DMA send and receive.
  EM1_UART0_HCR0[1:0] = 00b;

About how to disable flow control and FIFO, please refer to the process (6) of “**Chapter 3.3.2 Operation Detail**” of this document.

Appendix A UART Driver Function

A.1 UART Driver Function List

The following table shows the UART driver interface functions:

Table A-1 UART Driver Function List

Class	Function Name	Function Detail
Driver Function	em1_uart_init;	uart initialize.
	em1_uart_set_config;	set the UART configure.
	em1_uart_get_config;	get the uart configure.
	em1_uart_send;	send a character.
	em1_uart_recv;	receive a character.
	_em1_uart_enable_flw_cnt;	enable uart hardware flow control.
	em1_uart_disable_flw_cnt;	disable uart hardware flow control.
	_em1_uart_enable_fifo;	enale uart FIFO.
	em1_uart_disable_fifo;	disable uart FIFO.
	em1_uart_port_gio;	initialize the gio for simulate flow control.
	em1_uart_gio_send;	send a character with simulate flow control.
	em1_uart_gio_recv;	receive a character wiht simulate flow control.
	em1_uart_clear_error;	clear the receive error.
	em1_uart_print_error;	printf the receive error.
	em1_uart_print_configure;	printf the uart configure setting.
	em1_uart_port_uart;	initialize the uart port for flow control.
	em1_uart_enable_gio_recv;	enable GIO86 for receive.
	em1_uart_dma_enable;	enable UART for DMA.
	em1_uart_dma_disable;	disable UART for DMA.

A.2 UART Global Variable Define

The following table shows the UART global variable define:

Table A-2 Global Variable Define

Name	Type	Detail
g_uart_cts	volatile uchar	simulate cts is enable or not.

A.3 UART Structure Define

The following table shows the UART structure define:

Table A-3 Structure Define

Structure Name	Detail
struct st_UART_SETTING	UART setting information.

A.3.1 st_UART_SETTING

Table A-4 Structure of st_UART_SETTING

Member	Type	Detail
uart_num	uchar	the uart number.
uart_baudrate	uint	the uart baud rate.
uart_length	ushort	the transmit/receive bit numbers.
uart_parity	ushort	none, odd, even, stick high, stick low.
uart_stop	ushort	none, 1, 2 stop bit..
uart_mode	uchar	automatic flow control or not.
uart_triger	uchar	FIFO triger number.

A.4 UART Driver Function Detail

A.4.1 Initialize

[Function Name]

em1_uart_init

[Format]

DRV_RESULT em1_uart_init (uchar num);

[Argument]

Parameter	Type	I/O	Detail
num	uchar	I	UART number (0,1 or 2)

[Function Return]

DRV_OK: The function executes successfully.

DRV_ERR_PARAM: The input parameter is error.

[Flow Chart]

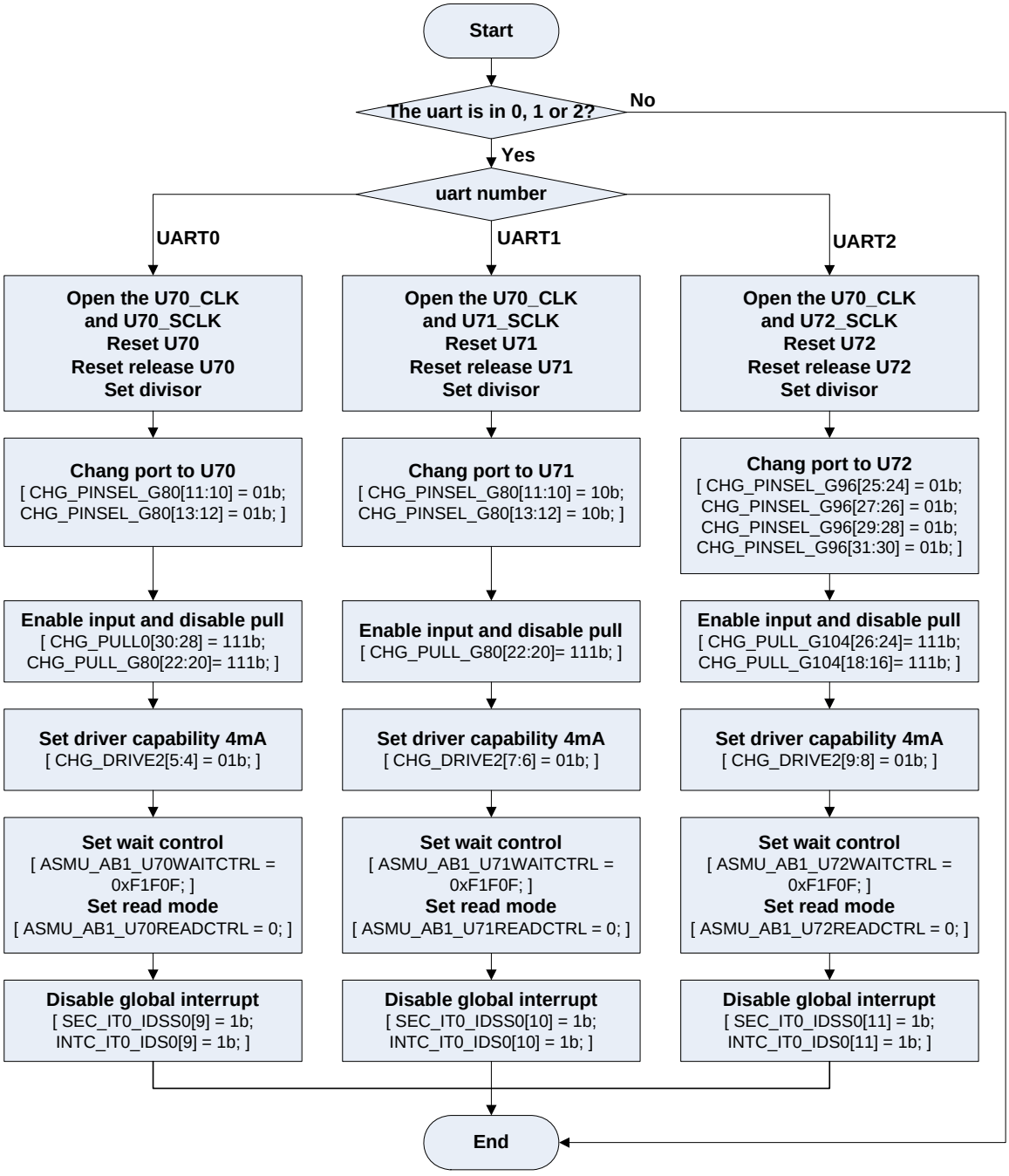

Figure A-1 UART Initialize

[Note]

(1). Open the clock.
The related registers are as follow:
ASMU_GCLKCTRL2ENA;
ASMU_GCLKCTRL2;

(2). Reset and reset release setting.
The related registers are as follow:
ASMU_RESETREQ0ENA;
ASMU_RESETCTRL0;

(3). Set divisor.
The related registers are as follow:
ASMU_DIVU70SCLK;
ASMU_DIVU71SCLK;
ASMU_DIVU72SCLK;

A.4.2 Set the Configure

[Function Name]

em1_uart_set_config

[Format]

DRV_RESULT em1_uart_set_config (struct st_UART_SETTING uart_value);

[Argument]

Parameter	Type	I/O	Detail
uart_value	struct st_UART_SETTING	I	UART information

[Function Return]

DRV_OK: The function executes successfully.

DRV_ERR_PARAM: The input parameter is error.

[Flow Chart]

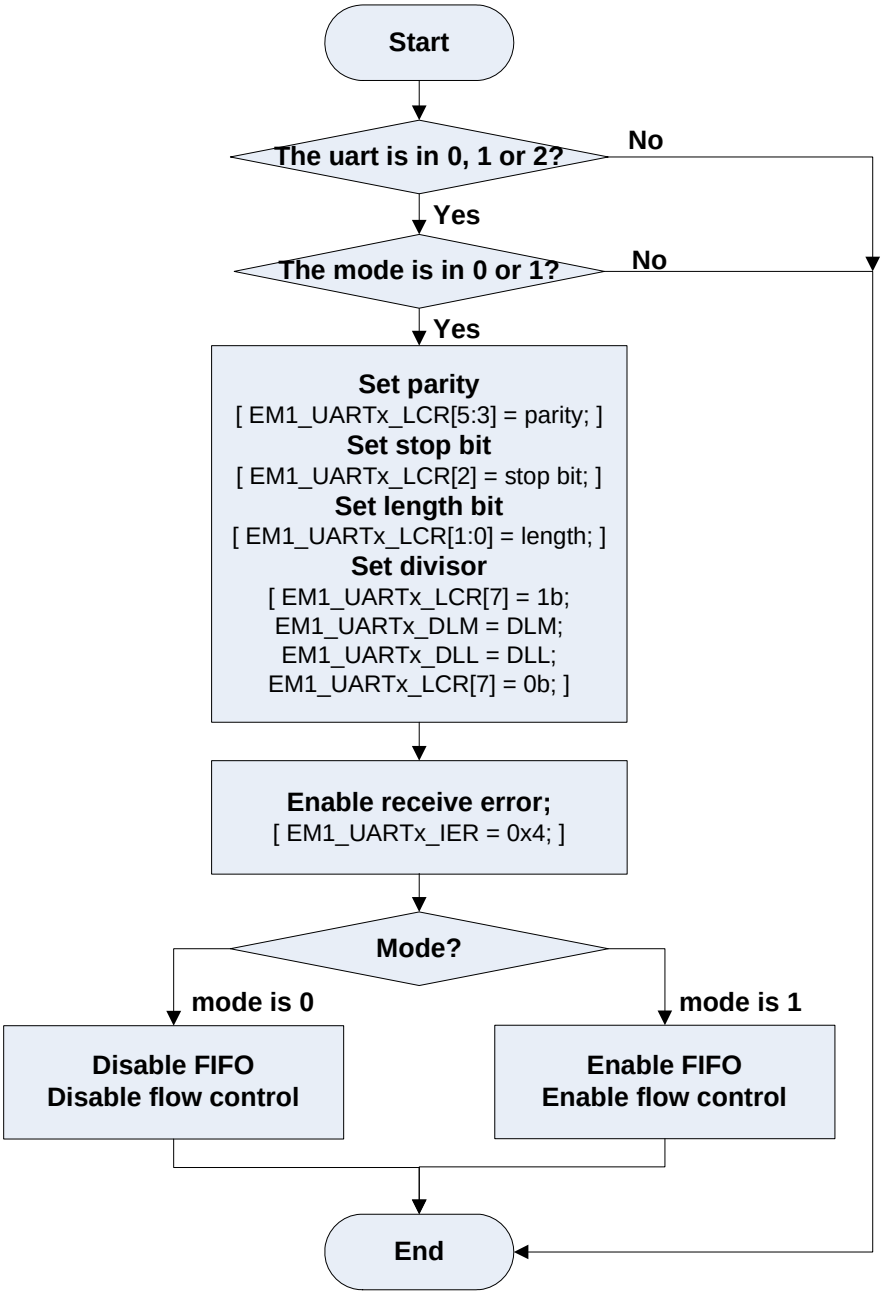

Figure A-2 Set the Configure

Note: x is 0, 1 or 2.

[Note]

(1). Check the input parameter.

If the UART number is not in 0, 1 or 2, the input parameter is error.

If the UART mode is not in 0 or 1, the input parameter is error.

(2). Set the UART attribute (**Note:** x is 0, 1, or 2).

```
// Set parity;
EM1_UARTx_LCR[5:3] = parity;
//Set stop bit;
EM1_UARTx_LCR[2] = stop bit;
// Set length bit;
EM1_UARTx_LCR[1:0] = length;
// Set divisor;
EM1_UARTx_LCR[7] = 1b;
EM1_UARTx_DLM = DLM;
EM1_UARTx_DLL = DLL;
EM1_UARTx_LCR[7] = 0b;
// Enable the UART receive error;
EM1_UARTx_IER = 0x4;
```

About how to set the parity, stop bit, length and divisor, please refer to the "**Chapter 3 REGISTERS**" of EMMA Mobile1 UART user's manual.

(3). Set the FIFO and flow control.

If the mode is 0, disable FIFO and the automatic flow control.

If the model is 1, enable FIFO and the automatic flow control.

A.4.3 Get the Configure

[Function Name]

em1_uart_get_config

[Format]

DRV_RESULT em1_uart_get_config (struct st_UART_SETTING *p_uart_value);

[Argument]

Parameter	Type	I/O	Detail
p_uart_value	struct st_UART_SETTING *	O	UART information

[Function Return]

DRV_OK: The function executes successfully.

DRV_ERR_PARAM: The input parameter is error.

[Flow Chart]

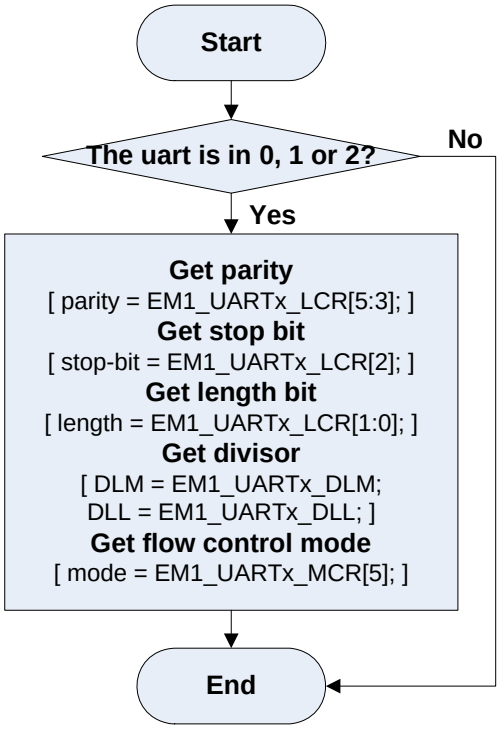

Figure A-3 Get the Configure

Note: x is 0, 1 or 2.

[Note]

(1). Check the input parameter.
If the UART number is not in 0, 1 or 2, the input parameter is error.

(2). Get the UART configure (**Note:** x is 0, 1, or 2).

```
// Get parity;
parity = EM1_UARTx_LCR[5:3];
// Get stop bit;
stop-bit = EM1_UARTx_LCR[2];
//Get length bit;
length = EM1_UARTx_LCR[1:0];
// Get divisor;
DLM = EM1_UARTx_DLM;
DLL = EM1_UARTx_DLL;
//Get flow control;
mode = EM1_UARTx_MCR[5];
```

A.4.4 Send a Character

[Function Name]

em1_uart_send

[Format]

DRV_RESULT em1_uart_send (uchar num, uchar c);

[Argument]

Parameter	Type	I/O	Detail
num	uchar	I	UART number
c	uchar	I	The character

[Function Return]

DRV_OK: The function executes successfully.

DRV_ERR_PARAM: The input parameter is error.

[Flow Chart]

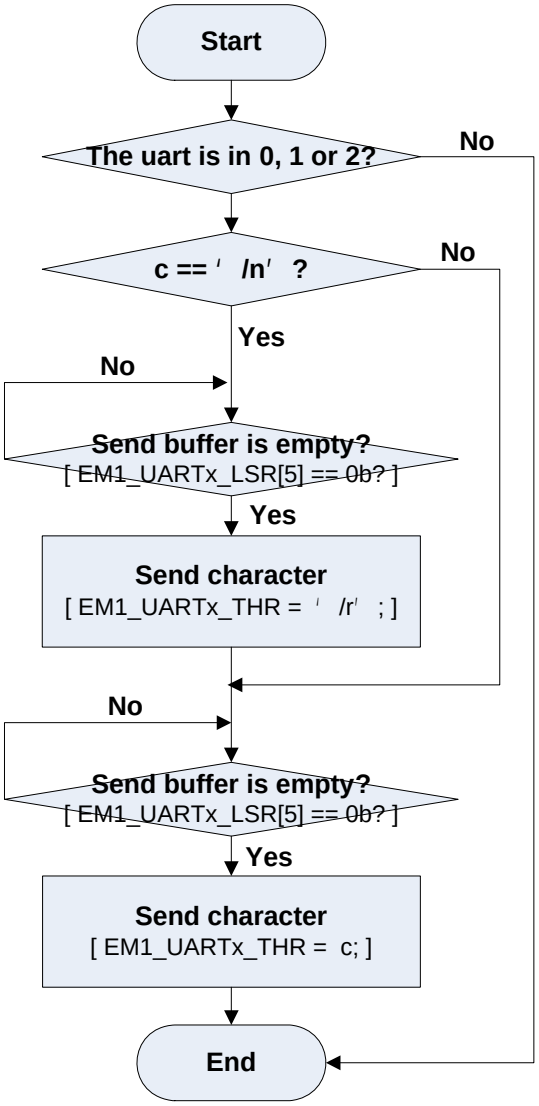

Figure A-4 Send a Character

Note: x is 0, 1 or 2.

[Note]

(1). Check the input parameter.
If the UART number is not in 0, 1 or 2, the input parameter is error.

(2). When the character is '\n', send '\r' (**Note:** x is 0, 1, or 2).
If EM1_UARTx_LSR[5] is 1, write the '\r' to EM1_UARTx_THR register.

(3). Send the character.
If EM1_UARTx_LSR[5] is 1, write the character to EM1_UARTx_THR register.

A.4.5 Receive a Character

[Function Name]

em1_uart_recv

[Format]

uchar em1_uart_recv (uchar num);

[Argument]

Parameter	Type	I/O	Detail
num	uchar	I	UART number

[Function Return]

DRV_ERR_PARAM: The input parameter is error.

Other: The received character.

[Flow Chart]

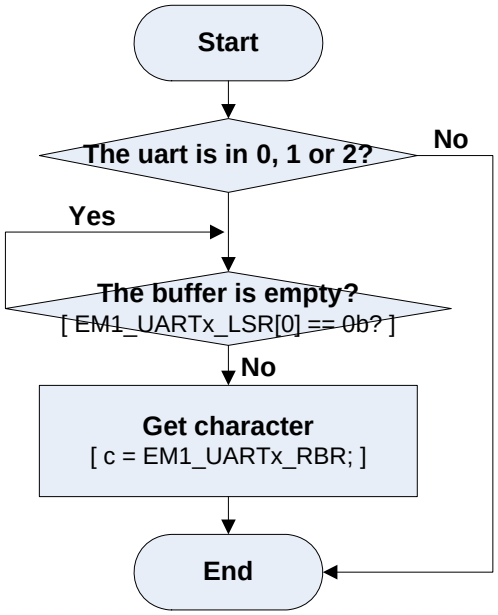

Figure A-5 Receive a Character

Note: x is 0, 1 or 2.

[Note]

(1). Check the input parameter.
If the UART number is not in 0, 1 or 2, the input parameter is error.

(2). Receive the character (**Note:** x is 0, 1, or 2).
If EM1_UARTx_LSR[0] is 1, get the character from EM1_UARTx_RBR register.

A.4.6 Enable the Flow Control

[Function Name]

_em1_uart_enable_flw_cnt

[Format]

DRV_RESULT _em1_uart_enable_flw_cnt (struct st_UART_SETTING uart_value);

[Argument]

Parameter	Type	I/O	Detail
uart_value	struct st_UART_SETTING	I	UART Information

[Function Return]

DRV_OK: The function executes successfully.

DRV_ERR_PARAM: The input parameter is error.

[Flow Chart]

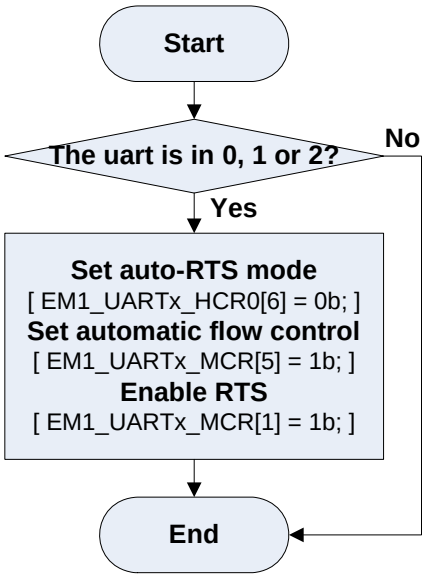

Figure A-6 Enable Flow Control

Note: x is 0, 1 or 2.

[Note]

(1). Check the input parameter.
If the UART number is not in 0, 1 or 2, the input parameter is error.

(2). Set the RTS mode (**Note:** x is 0, 1, or 2).
// Set RTS mode 0. (If use RTS mode 1, set 1 to EM1_UART0_HCR0[6]).
EM1_UART0_HCR0[6] = 0b;

About the difference between mode 0 and mode1, please refer to the "**Chapter 3.2.12 Hardware control register**" of EMMA Mobile1 UART user's manual.

(3). Enable automatic RTS and CTS.

```
// Enable the automatic flow control.
EM1_UARTx_MCR[5] = 1b;
// Enable RTS.
EM1_UARTx_MCR[1] = 1b;
```

A.4.7 Disable the Flow Control

[Function Name]

em1_uart_disable_flw_cnt

[Format]

DRV_RESULT em1_uart_disable_flw_cnt (uchar num);

[Argument]

Parameter	Type	I/O	Detail
num	uchar	I	UART number

[Function Return]

DRV_OK: The function executes successfully.

DRV_ERR_PARAM: The input parameter is error.

[Flow Chart]

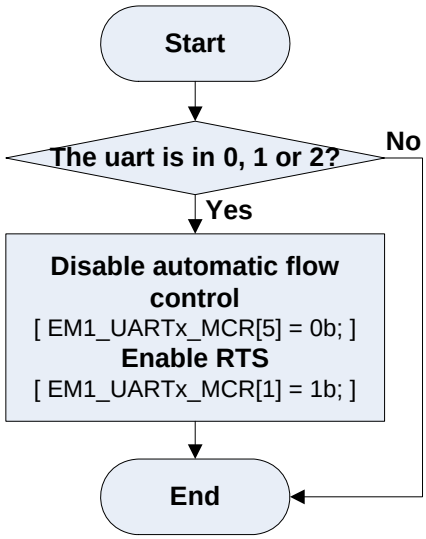

Figure A-7 Disable Flow Control

Note: x is 0, 1 or 2.

[Note]

(1). Check the input parameter.
If the UART number is not in 0, 1 or 2, the input parameter is error.

(2). Disable automatic RTS and CTS (**Note:** x is 0, 1, or 2).
// Disable the automatic flow control.
EM1_UARTx_MCR[5] = 0b;
// Enable RTS.
EM1_UARTx_MCR[1] = 1b;

A.4.8 Enable FIFO

[Function Name]

_em1_uart_enable_fifo

[Format]

DRV_RESULT _em1_uart_enable_fifo (struct st_UART_SETTING uart_value);

[Argument]

Parameter	Type	I/O	Detail
uart_value	struct st_UART_SETTING	I	UART information

[Function Return]

DRV_OK: The function executes successfully.

DRV_ERR_PARAM: The input parameter is error.

[Flow Chart]

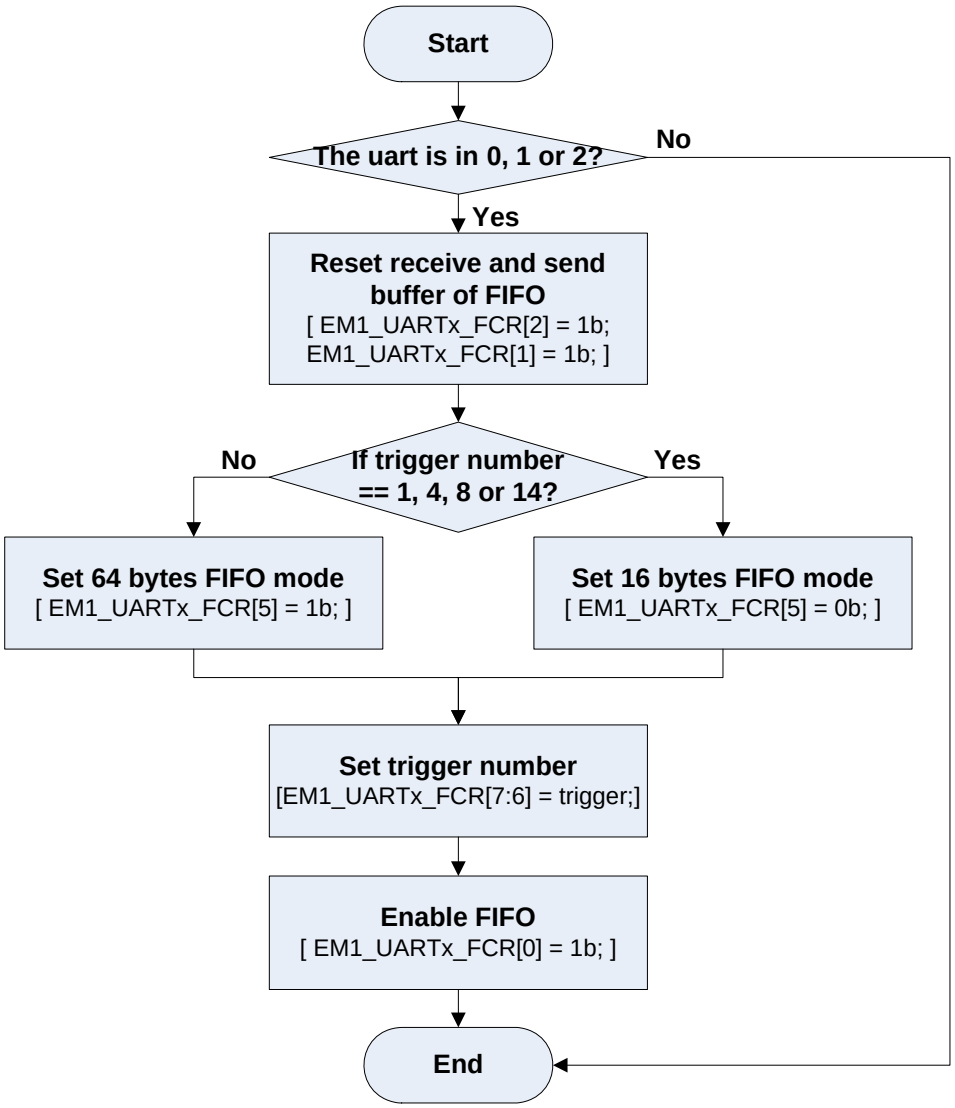

Figure A-8 Enable FIFO

Note: x is 0, 1 or 2.

[Note]

(1). Check the input parameter.

If the UART number is not in 0, 1 or 2, the input parameter is error.

(2). Reset the buffer of FIFO (**Note:** x is 0, 1, or 2).

// Reset the transfer buffer of FIFO.

EM1_UARTx_FCR[2] = 1b;

// Reset the receive buffer of FIFO.

EM1_UARTx_FCR[1] = 1b;

(3). Set the 16 bytes or 64 bytes FIFO mode.
// If the trigger number is 1, 4, 8 or 14, set 16 bytes FIFO mode.
EM1_UARTx_FCR[5] = 0b;
// If the trigger number is 32 or 56 bytes FIFO mode.
EM1_UARTx_FCR[5] = 1b;

(4). Set trigger number.
EM1_UARTx_FCR[7:6] = trigger number;

(5). Enable FIFO.
EM1_UARTx_FCR[0] = 1b;

A.4.9 Disable FIFO

[Function Name]

em1_uart_disable_fifo

[Format]

DRV_RESULT em1_uart_disable_fifo (uchar num);

[Argument]

Parameter	Type	I/O	Detail
num	uchar	I	UART number

[Function Return]

DRV_OK: The function executes successfully.

DRV_ERR_PARAM: The input parameter is error.

[Flow Chart]

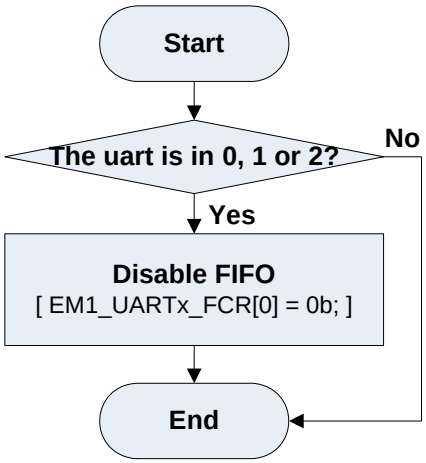

Figure A-9 Disable FIFO

Note: x is 0, 1 or 2.

[Note]

(1). Check the input parameter.
If the UART number is not in 0, 1 or 2, the input parameter is error.

(2). Disable FIFO (**Note:** x is 0, 1, or 2).
EM1_UARTx_FCR[0] = 0b;

A.4.10 Initialize GIO for Simulate Flow Control

[Function Name]

em1_uart_port_gio

[Format]

DRV_RESULT em1_uart_port_gio (void);

[Argument]

None

[Function Return]

DRV_OK: The function executes successfully.

DRV_ERR_PARAM: The input parameter is error.

[Flow Chart]

None.

A.4.11 Send a Character with Simulate Flow Control

[Function Name]

em1_uart_gio_send

[Format]

void em1_uart_gio_send (uchar c);

[Argument]

Parameter	Type	I/O	Detail
c	uchar	I	The character

[Function Return]

None.

[Flow Chart]

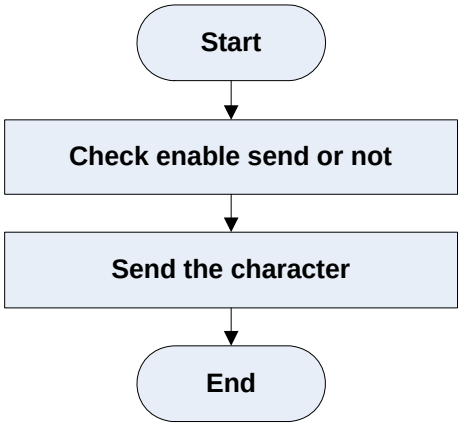

Figure A-10 Send a Character with Simulate Flow Control

[Note]

(1). Check CTS is enable or not.

```
// If the bit 85 is high, the received terminal can not receive characters.
// If the bit 85 is low, the received terminal can receive characters.
#define UART0_CTS_CHECK    \
do {      \
          if the GIO85 is low.        \
                    g_uart_cts = 0;  \
          else      \
                    g_uart_cts = 1;  \
} while ( g_uart_cts == 1 );
```

(2). Send the character.

```
em1_uart_send (0, c);
```

Remark: This function only uses GIO85 for UART0.

A.4.12 Receive a Character with Simulate Flow Control

[Function Name]

em1_uart_gio_recv

[Format]

uchar em1_uart_gio_recv ( );

[Argument]

None

[Function Return]

The received character.

[Flow Chart]

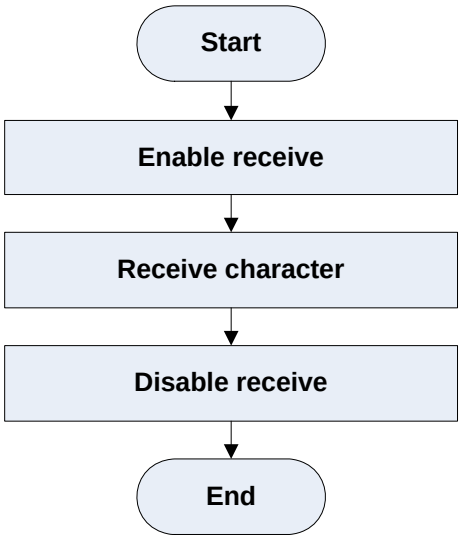

Figure A-11 Received a Character with Simulate Flow Control

[Note]

(1). Enable receive characters.
Set the port 86 low to enable the terminal send data.

(2). Receive a character.
c = em1_uart_recv (0);

(3). Disable receive characters.
Set the port 86 high to disable the terminal send data.

Remark:

This function only uses GIO86 for UART0.

A.4.13 Clear the Receive Error

[Function Name]

em1_uart_clear_error

[Format]

DRV_RESULT em1_uart_clear_error (uchar num);

[Argument]

Parameter	Type	I/O	Detail
num	uchar	I	UART number

[Function Return]

DRV_OK: The function executes successfully.

DRV_ERR_PARAM: The input parameter is error.

[Flow Chart]

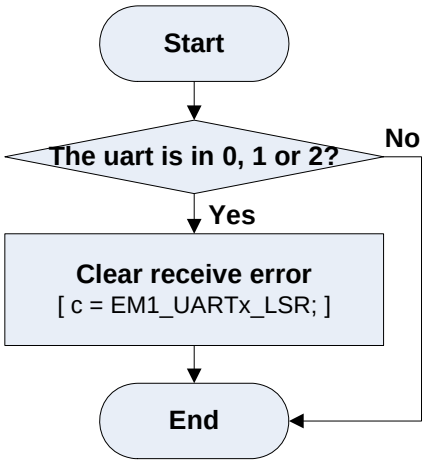

Figure A-12 Clear the Receive Error

Note: x is 0, 1 or 2.

[Note]

(1). Check the input parameter.
If the UART number is not in 0, 1 or 2, the input parameter is error.

(2). Clear the receive error (**Note:** x is 0, 1, or 2).
Read the EM1_UARTx_LSR to clear the receive error.

A.4.14 Print the Receive Error

[Function Name]

em1_uart_print_error

[Format]

DRV_RESULT em1_uart_print_error (uchar num);

[Argument]

Parameter	Type	I/O	Detail
num	uchar	I	UART number

[Function Return]

DRV_OK: The function executes successfully.

DRV_ERR_PARAM: The input parameter is error.

DRV_ERR_ABNORMAL: There is at least one error in receive process.

[Flow Chart]

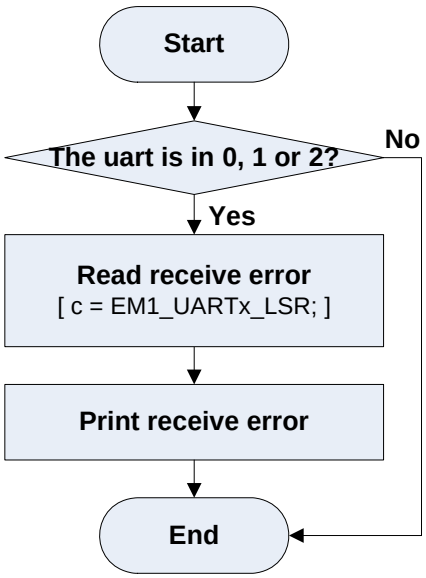

Figure A-13 Print the Receive Error

Note: x is 0, 1 or 2.

[Note]

(1). Check the input parameter.
If the UART number is not in 0, 1 or 2, the input parameter is error.

(2). Read the receive error (**Note:** x is 0, 1, or 2).
Read the receive error from EM1_UARTx_LSR.
(3). Print the error information.

A.4.15 Print the Configure

[Function Name]

em1_uart_print_configure

[Format]

void em1_uart_print_configure (struct st_UART_SETTING uart_value);

[Argument]

Parameter	Type	I/O	Detail
uart_value	struct st_UART_SETTING	I	UART Information

[Function Return]

None.

[Flow Chart]

None.

[Note]

None.

A.4.16 Initialize Port for Flow Control

[Function Name]

em1_uart_port_uart

[Format]

void em1_uart_port_uart (void);

[Argument]

None.

[Function Return]

None.

[Flow Chart]

None.

[Note]

None.

A.4.17 Enable GIO for Receive

[Function Name]

em1_uart_enable_gio_recv

[Format]

void em1_uart_enable_gio_recv (void);

[Argument]

None.

[Function Return]

None.

[Flow Chart]

None.

[Note]

Output low level from GIO86.

A.4.18 Enable UART DMA

[Function Name]

em1_uart_dma_enable

[Format]

DRV_RESULT em1_uart_dma_enable(uchar num, uchar send_recv);

[Argument]

Parameter	Type	I/O	Detail
num	uchar	I	UART number
send_recv	uchar	I	send and receive

[Function Return]

DRV_OK: The function executes successfully.

DRV_ERR_PARAM: The input parameter is error.

Flow Chart]

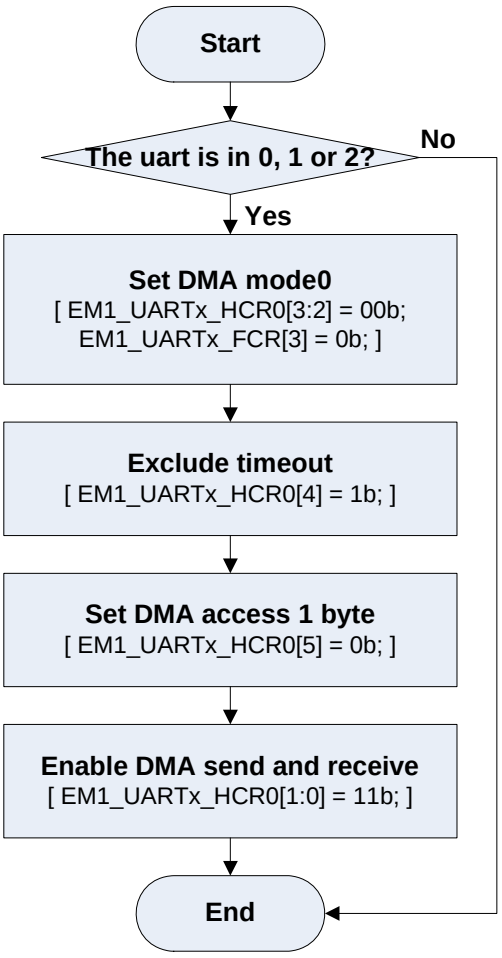

Figure A-14 Enable UART DMA

Note: x is 0, 1 or 2.

[Note]

(1). Check the input parameter.
If the UART number is not in 0, 1 or 2, the input parameter is error.

A.4.19 Disable UART DMA

[Function Name]

em1_uart_dma_disable

[Format]

DRV_RESULT em1_uart_dma_disable(uchar num, uchar send_recv);

[Argument]

Parameter	Type	I/O	Detail
num	uchar	I	UART number
send_recv	uchar	I	send and receive

[Function Return]

DRV_OK: The function executes successfully.

DRV_ERR_PARAM: The input parameter is error.

[Flow Chart]

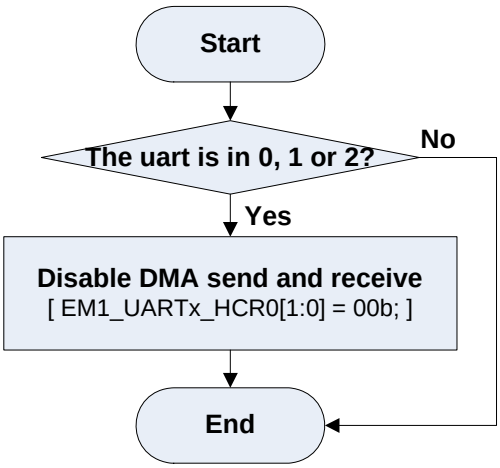

Figure A-15 Disable UART DMA

Note: x is 0, 1 or 2.

[Note]

(1). Check the input parameter.

If the UART number is not in 0, 1 or 2, the input parameter is error.

ANNEX Modification History

Number	Modification Contents	Author	Date
Ver 1.00	New version		Aug.4.2009